ชัยรัตน์ โล้วโสภณกุล

เทคนิคการแสดงผลใน เท็กซ์โหมด

นำเสนอเทคนิคต่าง ๆ ที่จะนำมาประยุกต์ใช้ในการเขียนโปรแกรมสำหรับแสดงผลในเท็กซ์โหมด

หลายคนคงพอจะจำสีเขียวของจอโมโนโครมกันได้ ซึ่งในตอนนั้นการเขียนโปรแกรมที่ต้องการการแสดงผลด้วยรูปแบบตัวอักษรแตกต่างจากตัวอักษรที่อยู่ในรอมเป็นเรื่องค่อนข้างจะลำบากมากทีเดียว เนื่องจากอาจจะต้องประกอบด้วยฮาร์ดแวร์ที่ออกแบบมาเฉพาะร่วมด้วย หรือไม่เช่นนั้นก็ต้องเขียนให้ทำงานในกราฟิกโหมด เป็นต้น

ต่อมาเมื่อจอภาพแบบวีจีเอเข้ามามีบทบาทมากขึ้น โปรแกรมต่าง ๆ ที่ถูกพัฒนาตามขึ้นมาก็จะเน้นสีสันและลูกเล่นต่าง ๆ มากขึ้นตามไปด้วย เพราะคุณสมบัติของจอวีจีเอนอกจากจะให้สีสันที่ชวนมองมากกว่าจอแบบโมโนโครมแล้ว ยังมีคุณสมบัติอื่น ๆ ที่น่าสนใจอีก ในบทความนี้เราจะแสดงให้เห็นคุณสมบัติอย่างหนึ่งคือ Character Generator ซึ่งเป็นคุณสมบัติที่มีเฉพาะในจออีจีเอและจอวีจีเอเท่านั้น เนื่องจากได้มีบทความเกี่ยวกับเรื่องนี้มาบ้างแล้วในฉบับก่อน ๆ ทางผู้เขียนจึงขอกล่าวเพียงย่อ ๆ เพื่อเป็นการทบทวนความจำ และเป็นการสร้างความเข้าใจให้กับผู้อ่านที่ยังไม่ทราบในส่วนของการสร้างฟอนต์ และรายละเอียดเกี่ยวกับอินเตอร์รัพต์

อินเตอร์รัพต์ที่นำมาใช้งาน

หัวใจของการทำงานจะอยู่ที่อินเตอร์รัพต์ 10h ฟังก์ชัน 11h ของไบออส ซึ่งรายละเอียดต่าง ๆ รวมไปถึงฟังก์ชันย่อย (SubFunction) ได้แสดงไว้ดังรูปที่ 1

จากรูปจะเห็นได้ว่าฟังก์ชันย่อยที่เราเลือกใช้ในโปรแกรมมี 2 ฟังก์ชันย่อย คือ ฟังก์ชันย่อยที่ 00 (Function Load User Text Font) และฟังก์ชันย่อยที่ 04 (Function Load ROM 8X16 Text Font) ซึ่งทั้ง 2 ฟังก์ชันย่อยนี้ใช้งานแตกต่างกันคือ ฟังก์ชันย่อยที่ 00 จะใช้สำหรับโหลดฟอนต์ที่เราออกแบบไว้ไปใช้แทนฟอนต์ปกติจากรอม และฟังก์ชันย่อยที่ 04 จะใช้ในทางกลับกันคือ เมื่อเราต้องการจะเปลี่ยนกลับไปใช้ฟอนต์ปกติจากรอม

การออกแบบฟอนต์

การออกแบบฟอนต์ในบทความนี้จะเลือกใช้ฟอนต์ขนาด 8X16 พิกเซล (Pixcel) กล่าวคือ ตัวอักษร 1 ตัว จะมีความกว้าง 8 พิกเซล และมีความสูง 16 พิกเซล เพราะว่าเป็นฟอนต์ที่มีขนาดคุ้นเคยกันดี อีกทั้งยังสะดวกที่จะออกแบบให้ใช้กับตัวอักขระไทยอีกด้วย

การสร้างฟอนต์

ขั้นแรกเรามาทำความเข้าใจเกี่ยวกับการกำหนดค่าตัวเลขแทนจุดต่าง ๆ (ตัวเลขเป็นเลขฐาน 16) ในแต่ละแถวจะมี 8 ช่อง ซึ่งเราสามารถคำนวณหาค่าแทนจุดของแต่ละแถวได้ดังนี้

ค่าประจำจุด	128	64	32	16	8	4	2	1
	0	0	0			0	0	0

Register	High	Low
AX	11h	Subfunction
BX	*	*
CX	*	*
DX	*	*

SP	
BP	Offset of pointer to user table *
SI	
DI	

IP	
Flags	

CS	
DS	
SS	
ES	Segment of pointer to user table *

รูปที่ 1 Interrupt 10h Function 11h Character Generator
ค่าที่ต้องกำหนดให้รีจิสเตอร์ต่าง ๆ

AL Value	Other Registers Used
00=load user text font	BH=number of bytes per char BL=block CX=number of chars DX=ASCII of 1st character ES:BP=pointer to table
01=load ROM 8x14 text font	BL=block to load
02=load ROM 8x8 text font	BL=block to load
03=set block specifier	BL=select character block
04=load ROM 8x16 text font(VGA)	BL=block to load
10h=load user text font	BH=number of bytes per char BL=block CX=number of chars DX=ASCII of 1st character ES:BP=pointer to table
11h=load ROM 8x14 text font	BL=block to load
12h=load ROM 8x8 text font	BL=block to load
14h=load ROM 8x16 text font(VGA)	BL=block to load
20h=set user graphics char pointer to INT 1Fh	ES:BP = pointer to user graphics font
21h=set user graphics char pointer to INT 43h	BL=rows(coded) CX=bytes per chars DL=rows per screen ES:BP = pointer to table
22h=use ROM 8x14 font for graphics	BL=rows(coded),DL=rows/screen
23h=use ROM 8x8 font for graphics	BL=rows(coded),DL=rows/screen
24h=use ROM 8x16 font for graphics	BL=rows(coded),DL=rows/screen
30h=get font pointer info	BH=font pointer (coded) Returns: CX=bytes per character DL=rows ES:bp=pointer

รูปที่ 1 (ต่อ) Sets Subfunction to perform

โดยเครื่องหมาย o แทนจุดที่ต้องการให้แสดง และช่องว่างแทนจุดที่ไม่ต้องการให้แสดง

จากรูปเราสามารถหาค่าตัวเลขที่ใช้แทนจุดของแถวนี้ได้ โดยเอาค่าประจำจุดของแต่ละตำแหน่งที่ต้องการให้แสดงมาบวกกัน แล้วแปลงกลับไปเป็นเลขฐานสิบหกจากตัวอย่างดังนี้

```
128 + 64 + 32 + 4 + 2 + 1 = 231 (ในฐานสิบ)
```

แล้วทำการแปลงไปเป็นเลขฐานสิบหกก็จะได้

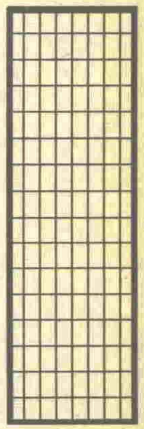
```
231d = E7h
```

ในรูปที่ 2 เราจะเห็นได้ว่าในตัวอักษรหนึ่งตัวจะประกอบด้วยแถวทั้งหมด 16 แถว เรียงกันไป จึงทำให้ตัวอักษรหนึ่งตัวต่อที่ใช้เลขแทนจุดนี้ทั้งหมด 16 ชุด (หรือ 16 ไบต์)

เพื่อความสะดวก ทางผู้เขียนจึงได้เตรียมฟอนต์แบบต่าง ๆ ไว้ให้ผู้อ่านนำไปใช้ประยุกต์กับโปรแกรมได้เลย ซึ่งจะมีทั้งฟอนต์ตัวอักษรภาษาอังกฤษรวมไปถึงฟอนต์ภาษาไทยด้วย โดยจะเก็บไว้ในไฟล์ซึ่งมีนามสกุล XXXXXXXX .FNT ขนาดของไฟล์คำนวณได้จาก

```
ขนาดของไฟล์ = จำนวนตัวอักษร x 16
```

เช่น ไฟล์ที่เก็บตัวอักษร 256 ตัว จะมีขนาด 256 x 16 = 4,096 ไบต์ เป็นต้น



รูปที่ 2 โครงสร้างของหนึ่งตัวอักษร ซึ่งจะมีขนาด 8 x 16 Pixcels

การทำงาน

จากลิสติงเราจะกำหนดตัวแปร BytesPerChar ซึ่งใช้เก็บค่าของจำนวนไบต์ต่อหนึ่งตัวอักษรให้เป็นค่าคงที่ มีค่าเท่ากับ 16 เพราะว่าหากมีการเปลี่ยนโหมดการแสดงผลตัวอักษรที่ใช้ก็จะมีจำนวนไบต์แตกต่างกันออกไป จึงเป็นการสะดวกในการแก้ไขในภายหลัง ส่วนตัวแปร FontType เป็นตัวแปรชนิดอะเรย์ (Array) ประกอบด้วยอะเรย์ของตัวอักษรแต่ละตัวที่มีขนาดเท่ากับ BytesPerChar จำนวน 256 ตัว

ลิสติงที่ 1

```
PROGRAM Font_Demo;

(*********************************************************)
(*   Program    :  Font Demo                             *)
(*   Purpose    :  Display New Techincs Font in Text Mode  *)
(*   Compiler   :  Turbo Pascal V.6.0                    *)
(*********************************************************)

USES Crt,Dos;

CONST BytesPerChar = 16;

TYPE  OneChar       = ARRAY[1..BytesPerChar] OF BYTE;
      FontType      = ARRAY[0..255] OF OneChar;

VAR   Regs          : REGISTERS;
      Font          : FontType;
      FontFile      : FILE OF FontType;
      FontFileName  : STRING;
      Path          : PathStr;
      Dir           : DirStr;
      Name          : NameStr;
      Ext           : ExtStr;
      OneC          : OneChar;
      C             : CHAR;

PROCEDURE LoadOneChar(Number : BYTE;  VAR C : OneChar);
BEGIN
      Font[Number] := C;
      FillChar(Regs,SizeOf(Regs),0);
      Regs.AH := $11;
      Regs.AL := $00;
      Regs.BL := 0;
      Regs.BH := BytesPerChar;                  { bytes pr character            }
      Regs.ES := Seg(Font);                     { segment address of font table }
      Regs.BP := Ofs(Font)+BytesPerChar*Number;{ offset address of font table  }
      Regs.CX := 1;                             { number of characters to load  }
      Regs.DX := Number;                        { table character offset        }
      Intr($10,Regs);
END;

PROCEDURE LoadUserFont;
BEGIN
      FillChar(Regs,SizeOf(Regs),0);
      Regs.AH := $11;
      Regs.AL := $00;
      Regs.BL := 0;
      Regs.BH := BytesPerChar;                  { bytes pr character            }
      Regs.ES := Seg(Font);                     { segment address of font table }
      Regs.BP := Ofs(Font);                     { offset address of font table  }
      Regs.CX := $100;                          { number of characters to load  }
      Regs.DX := 0;                             { table character offset        }
      Intr($10,Regs);
END;

PROCEDURE ClearUserFont;
BEGIN
      FillChar(Regs,SizeOf(Regs),0);
      Regs.AH := $11;
      Regs.AL := $04;
      Regs.BL := 0;
      Regs.BH := BytesPerChar;                  { bytes pr character            }
      Regs.ES := Seg(Font);                     { segment address of font table }
      Regs.BP := Ofs(Font);                     { offset address of font table  }
      Regs.CX := 0;                             { number of characters to load  }
      Regs.DX := 0;                             { table character offset        }
      Intr($10,Regs);
END;

PROCEDURE InitChar;
VAR   I : BYTE;
BEGIN
      FOR I:= 1 TO 16 DO
        OneC[I] := $C3;
END;
```

→

ลิสติงที่ 1 (ต่อ)

```
PROCEDURE DisplayFont;
VAR   I : Byte;
BEGIN
      TextAttr := Red SHL 4 + Yellow;
      Window(19,5,60,12);
      ClrScr;
      Window(1,1,80,25);
      FOR I := 40 TO 255 DO
         BEGIN
         GoToXY(20+(I MOD 40),5+(I DIV 40));
         Write(Chr(I));
      END;
END;

PROCEDURE BackGround;
VAR I,J : BYTE;
BEGIN
      TextAttr := Blue SHL 4 + White;
      GoToXY(1,2);
      FOR I := 1 TO 23 DO
         BEGIN
            Write(' ');
            Write(' ');
         END;
      TextAttr := Red SHL 4 + Yellow;
      DisplayFont;
END;

PROCEDURE RunDemo1;
VAR I : BYTE;
BEGIN
      GoToXY(26,15);
      Write('DEMO 1 : Press any key...');
      I := 0;
      REPEAT
          Inc(I);
          CASE I OF
              1 : OneC[8] := $AA;      2 : OneC[8] := $EA;
              3 : OneC[8] := $FA;      4 : OneC[8] := $FE;
              5 : OneC[8] := $FF;      6 : OneC[8] := $BF;
              7 : OneC[8] := $AF;      8 : OneC[8] := $AB;
          END;
          LoadOneChar(138,OneC);
          Delay(200);
          IF I>=8 THEN I:= 0;
      UNTIL KeyPressed;
      C := ReadKey;
END;

PROCEDURE RunDemo2;
VAR I : BYTE;
BEGIN
      GoToXY(26,15);
      Write('DEMO 2 : Press any key...');
      I := 0;
      REPEAT
         Inc(I);
         OneC[1+Random(16)] := Random($FF);
         LoadOneChar(138,OneC);
         Delay(150);
         IF I>=8 THEN I:= 0;
      UNTIL KeyPressed;
      C := ReadKey;
END;

FUNCTION ReadFontFile(FontFileName : STRING) : BOOLEAN;
BEGIN
  {$I-}
      Assign(FontFile,FontFileName);
      ReSet(FontFile);
  {$I+}
      IF IOResult=0 THEN
```

→

ลิสติงที่ 1 (ต่อ)

```
            BEGIN
                Read(FontFile,Font);
                Close(FontFile);
                ReadFontFile := TRUE;
            END
        ELSE ReadFontFile := FALSE;
END;

BEGIN
        WriteLn('Font Demo');
        WriteLn;
        IF ParamCount<>1 THEN
            BEGIN
                WriteLn('Wrong number of parameters.');
                WriteLn('Usage : LOAD filename.ext');
                Halt(1);
            END;
        Path := ParamStr(1);
        FSplit(Path,Dir,Name,Ext);
        IF Ext='' THEN
                Path := Dir+Name+'.FNT';
        IF ReadFontFile(Path) THEN
            BEGIN
                LoadUserFont;
                WriteLn('Font ',Name,' loaded.');
            END
        ELSE    WriteLn('Error loading fontfile ',Path);
        WriteLn;
        WriteLn('Press any Key ...');
        C := ReadKey;
        ClrScr;
        DisplayFont;
        C := ReadKey;
        ClrScr;
        BackGround;
        InitChar;
        RunDemo1;
        RunDemo2;
        TextAttr := Black SHL 4 + LightGray;
        ClrScr;
        ClearUserFont;
END.
```

ในโพรซีเดอร์ LoadOneChar ใช้ในการโหลดตัวอักษรที่ผู้ใช้ออกแบบไว้เพียงหนึ่งตัว โดยตัวอักษรที่เหลือยังคงไม่เปลี่ยนแปลง ในการเรียกใช้จะประกอบด้วยพารามิเตอร์ 2 ตัว คือ หมายเลขแอสกี (ASCII) ของตัวอักษรที่ต้องการโหลดกับตัวแปรซึ่งเก็บค่ารหัสแทนจุดของตัวอักษรนั้น และโพรซีเดอร์ LoadUserFont ใช้ในการโหลดตัวอักษรที่ผู้ใช้ออกแบบไว้ทั้งหมด 256 ตัว ซึ่งทั้งสองโพรซีเดอร์นี้จะเป็นการเรียกใช้ฟังก์ชันที่ 11h ฟังก์ชันย่อยที่ 0 ของอินเตอร์รัพต์ที่ 10h เป็นการโหลดฟอนต์ของผู้ใช้ จำนวนตัวอักษรที่ต้องการโหลดจะถูกกำหนดโดยค่าในรีจิสเตอร์ CX และรีจิสเตอร์ DX ใช้ในการกำหนดค่ารหัสแอสกีตัวแรกของตัวอักษรที่ต้องการโหลด โดยมีรีจิสเตอร์ ES:BP กำหนดค่าของเซกเมนต์และออฟเซตของตารางข้อมูลตามลำดับ

โพรซีเดอร์ ClearUserFont ใช้ในการคืนฟอนต์กลับสู่ฟอนต์ปกติของเครื่อง โดยเรียกใช้ฟังก์ชันที่ 11h ฟังก์ชันย่อยที่ 4 ของอินเตอร์รัพต์ที่ 10h ซึ่งเป็นการโหลดฟอนต์จากรอมไบออส

โพรซีเดอร์ RunDemo1 จะแสดงตัวอย่างการนำเสนอในแบบแรก คือ แสดงแบคกราวนด์แบบมีจุดวิ่งอยู่ภายใน เริ่มต้นด้วยการกำหนดค่าแพตเทอร์นของการวิ่งในแต่ละครั้ง แล้วให้ทำการแสดงผลทีละขั้นต่อเนื่องกันไปจนกว่ามีการกดคีย์ใด ๆ

โพรซีเดอร์ RunDemo2 เป็นตัวอย่างการนำเสนอแบบที่สอง คือ การแสดงแบคกราวนด์แบบมีการสุ่มแพตเทอร์นภายในซึ่งจะเปลี่ยนรูปแบบไปเรื่อยจนกว่าจะมีการกดคีย์ใด ๆ

การนำไปประยุกต์ใช้งาน

ในบางกรณีหากผู้อ่านต้องการโหลดตัวอักษรเพียงแค่ 128 ตัวอักษร (เช่น ต้องการโหลดเฉพาะตัวอักษรภาษาไทย) ก็ให้ทำการแก้ตัวแปรแบบอะเรย์ (Array) เป็น FontType = ARRAY[127..255] OF OneChar และค่าที่ป้อนให้กับรีจิสเตอร์ CX กับรีจิสเตอร์ DX ในโพรซีเดอร์ LoadUserFont โดยแก้เป็น Regs.CX := $80 และ Regs.DX := $80 ตามลำดับ

ส่วนในโพรซีเดอร์อื่น ๆ นั้นใช้ประกอบในการแสดงลูกเล่นแบบต่าง ๆ ซึ่งพอจะทำให้ผู้อ่านเห็นว่าการแสดงผลในแบบเท็กซ์โหมดนั้นก็สามารถทำให้มีความสวยงามดึงดูดใจได้เช่นเดียวกัน

การทดสอบโปรแกรม

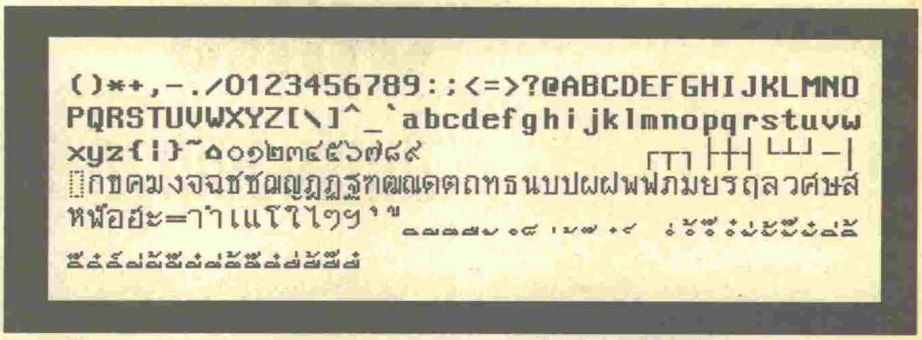
โปรแกรมทดสอบที่เขียนขึ้นนี้จะแสดงให้เห็น 3 แบบด้วยกัน คือ

1. แสดงตัวอักษรทั้งหมดที่ถูกโหลดขึ้นมา ดังแสดงในรูปที่ 3

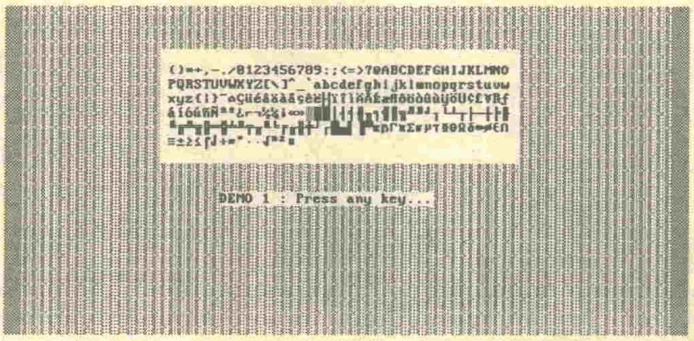
2. แสดงตัวอักษรทั้งหมดที่ถูกโหลด และแสดงตัวอย่างการทำแบคกราวนด์ ที่มีการเคลื่อนไหวภายในตัวอักษรที่ใช้ทำแบคกราวนด์ตลอดเวลา ดังรูปที่ 4

3. แสดงตัวอักษรทั้งหมดที่ถูกโหลด และแสดงตัวอย่างการทำแบคกราวนด์ที่มีการสุ่มรูปแบบของตัวอักษรที่ใช้ทำแบคกราวนด์ตลอดเวลา ดังแสดงในรูปที่ 5

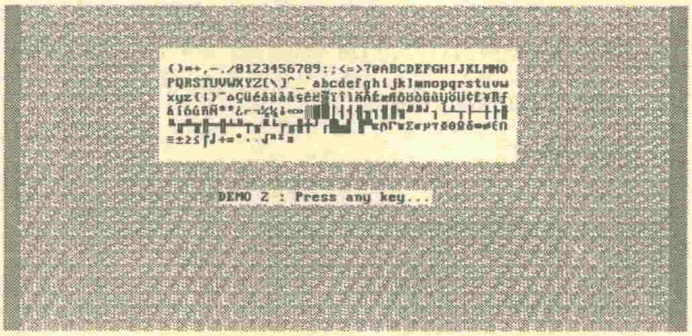
รูปที่ 3



รูปที่ 4



รูปที่ 5



ในการรันโปรแกรมนั้นทำได้โดยพิมพ์ชื่อโปรแกรมแล้วตามด้วยพารามิเตอร์ 1 ตัว ซึ่งเป็นชื่อฟอนต์ที่ต้องการโหลด ตัวอย่างเช่น

```
A:TEST NEWTHAI (นามสกุลจะใส่หรือไม่ก็ได้)
```

ฝากเอาไว้

จากเทคนิคที่ได้กล่าวมาแล้วข้างต้น ผู้อ่านคงพอที่จะได้แนวทางในการนำเสนอหรือนำไปประยุกต์ใช้กับการเขียนโปรแกรมให้มีความดึงดูดใจมากยิ่งขึ้น เช่น อย่างแรกก็คือใช้ความสามารถ พื้นฐานนี้ในการนำเสนอโปรแกรมในรูปแบบของตัวอักษรไทยมีการติดต่อทักทายเป็นภาษาไทย โดยในส่วนของการจัดสระหรือวรรณยุกต์นั้นได้มีผู้นำมาเขียนแสดงให้ดูกันบ้างพอสมควรแล้ว

ส่วนเทคนิคอื่น ๆ ที่ผู้เขียนนำไปประยุกต์ใช้ก็มีอีกหลายอย่าง เช่น ใช้ในการเขียนเกม ซึ่งทำงานในเท็กซ์โหมด (เช่น รูปทรงต่าง ๆ ของเกม Tetris) หรือใช้ในการเปลี่ยนเคอร์เซอร์ ซึ่งแต่เดิมเราจะเห็นได้ว่าในเท็กซ์โหมดนั้นการแสดงเคอร์เซอร์มีเพียงแค่เป็นแท่งหรือขีดเท่านั้น แต่ด้วยเทคนิคนี้เราสามารถที่จะแสดงเคอร์เซอร์เป็นรูปต่าง ๆ ได้ โดยทำการซ่อนเคอร์เซอร์เดิมเสียก่อน แล้วทำการอ่านค่าตำแหน่งปัจจุบันของเคอร์เซอร์ ต่อจากนั้นให้แสดงตัวอักษรที่ได้ออกแบบไว้ ณ ตำแหน่งเคอร์เซอร์ที่อ่านมาได้ โดยอาจจะเขียนโปรแกรมเป็นแบบฝังในหน่วยความจำก็ได้ ซึ่งพอจะยกตัวอย่างให้เห็นได้ง่าย ๆ ดังนี้ เช่น เมื่อเราต้องการให้ป้อนหมายเลขโทรศัพท์ก็ทำได้โดยเปลี่ยนเคอร์เซอร์เป็นรูปโทรศัพท์ เป็นต้น

สำหรับเทคนิคเหล่านี้ผู้อ่านคงพอที่จะมองเห็นแนวทางในการนำไปประยุกต์ให้โปรแกรมน่าสนใจมากยิ่งขึ้น ผู้เขียนหวังว่าประกายเล็ก ๆ เหล่านี้คงจะพอเป็นประโยชน์แก่ผู้อ่านบ้าง หากใครมีเทคนิคดี ๆ อย่าลืมแนะนำกันบ้าง สวัสดีครับ